BUNKER: a Blockchain-based trUsted VNF pacKagE Repository

Scheid, Eder John ; Keller, Manuel ; Franco, Muriel F ; Stiller, Burkhard

DOI: https://doi.org/10.1007/978-3-030-36027-6_16

BUNKER: a Blockchain-based trUsted VNF packagE Repository

Eder J. Scheid, Manuel Keller, Muriel F. Franco, and Burkhard Stiller

Communication Systems Group CSG, Department of Informatics IfI,
University of Zurich UZH, Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
{scheid, franco, stiller}@ifi.uzh.ch, manuel.keller@bf.uzh.ch

Abstract. Current projects applying blockchain technology to enhance the trust of NFV environments do not consider the VNF repository. However, the blockchain's properties can enhance trust by allowing to verify a VNF package's integrity without relying (a) on a Trusted Third Party (TTP) for remote attestation or (b) a secure database. This paper presents BUNKER, a Blockchain-based trUsted VNF packagE Repository, intended to be integrated with traditional database-based package verification environments, acting as a trusted repository containing VNF package information. Moreover, BUNKER allows users to acquire VNFs without the need of a TTP using an Ethereum Smart Contract (SC). The SC automatically transfers license fees to the vendor once a VNF is acquired, and sends the VNF package's link to the buyer before verifying its integrity.

Keywords: Network Functions Virtualization · Blockchain · Repository.

1 Introduction

The deployment of Network Functions Virtualization (NFV) [11] solutions faces a major challenge regarding the incorporation of trust to end-users. For example, with the myriad of novel Virtual Network Functions (VNF) being developed, it remains an open problem on how to ensure that the VNF package being acquired by end-users is not malicious and it was not tampered with. Research has been conducted in the NFV computing environment with the introduction of Trusted Platform Modules (TPM) and remote attestation services [12]. Although these systems are able to verify the state of the NFV environment, they rely on a central database to verify the VNF's package integrity. Thus, this centralization enforces end-users to trust in the repository holding VNF packages and presenting a single point of failure and a bottleneck.

Recent Blockchain (BC) developments focused on the provisioning of trust, including Smart Contracts (SC). The BC concept was first described in 2009 in the context of the cryptocurrency Bitcoin [10]. In general, a BC is a distributed ledger where each new appended block contains transactions and information (*i.e.*, the block hash) about the previous block. The most important properties of BCs are their data immutability and data decentralization [16]. The former

ensures that once data is included in the blockchain, it cannot be altered or removed; while the latter provides high data availability. These properties form the perfect environment for the execution of SCs. In Ethereum [4], SCs are written in a Turing-complete programming language, called Solidity [6]. This Turing-completeness allows for the creation of complex functions and helps to enforce a variety of contracts through cryptographic principles [1]. Solidity-based SCs can be used to facilitate trusted exchanges between untrusted entities and the correct execution of programmed SC code. These properties can be used in the context of NFV to address trust deficits regarding the VNF package integrity verification.

This paper presents the design of a blockchain-based trusted VNF package repository, called `BUNKER`, which provides trusted and immutable information concerning VNF packages acquired by end-users. Thus, end-users are not bound to trust on a central trusted authority, but rather on a distributed and highly available data source, *i.e.*, the BC. Moreover, `BUNKER` allows end-users to acquire VNF packages without the need of a Trusted Third Party (TTP) and automatically `BUNKER` transfers the license fee to the developer or vendor. To guarantee the integrity of the VNF package, `BUNKER` stores the hash of the VNF package so that end-users, after receiving the VNF package, can verify whether it had been tampered with. An implementation prototype of `BUNKER` is available at [9].

The remainder of this paper is organized as follows. Section 2 provides an overview of related work on existing VNF marketplaces and uses of the blockchain technology for management and orchestration in the NFV context. Section 3, presents the design of `BUNKER`, while Section 4 discusses open challenges. Section 5 summarizes the paper and outlooks on future work.

2 Related Work

Currently, marketplaces providing VNF-as-a-Service (VNFaaS) have been receiving attention. FENDE [2] is a Marketplace and a Federated Ecosystem for the Distribution and Execution of VNFs. It presents to the user the compatible VNFs currently listed in a traditional central database-based repository. In addition, FENDE includes NFV management and orchestration tools, which allow users to deploy and manage licensed services in the same ecosystem. T-Nova [16] enables network operators to virtualize their network functions as well as offer them to their clients in an on-demand, per-customer model. This model allows them to provide network services to their customers without having to deploy specialized hardware on the customer's premises. A traditional database-based marketplace is available for customers to acquire and instantiate their required network services on-demand.

BC is independent of any authorization entity and establishes trust between untrusted peers. Moreover, the immutability of public BCs determines a highly suitable feature in areas where audibility is crucial. Thus, these combined properties lead to research regarding the employment of BC in the NFV context. Virtual Machine Orchestration Authenticator (VMOA) [3] is an authentication

model that establishes a trustful Virtual Machine (VM) environment. Instead of having an internal or external trusted authenticator, [3] propose to establish a VMOA BC to offload the authentication responsibility to a distributed ledger. In this system, each orchestration request is sent to a BC, authenticated and only then sent to the virtualization server. If successful, the VM manager reports the success to the BC. Each step is stored in the BC and is auditable. The implementation applies a private BC based on the Hyperledger framework. [13] proposes a BC-based NFV Management and Orchestration (MANO) solution. SINFONIA (Secure vIrtual Network Function Orchestrator for Non-repudiation, Integrity, and Auditability) is designed for data centers in which multiple network services from different clients are deployed. The BC-based NFV architecture addresses all requirements. The prototype implementation shows that the proposed architecture ensures high availability and eliminates the single point of failure. However, it is not clear where the BC nodes are located and which are the incentives for peers to maintain these BC nodes.

There have been efforts in developing marketplaces for VNF packages [2,16]. These lead to the creation of systems where users can access a VNF repository containing various packages that can be deployed easily. However, they are centralized and require that the user trusts in the database solution of the provider. [3,13] address the trust challenge by incorporating BCs into the NFV MANO, while securing the computing environment and the configurations. However, they do not extend to the VNF repository or exploit benefits of relying on a public BC. This leads to a security flaw, where malicious actors can gain access to the central VNF repository to inject malicious code. Even though VNFs are executed in a secure environment, this compromises the security of the entire system. So far, research has shown that a trusted NFV environment should extend to the VNF package repository and that the properties of public BCs are promising to address such a gap. Further, none of the approaches as listed in Table 1 (where ✓ means addressed and ✗ means not addressed) address the combination of *(a)* BC and NFV, *(b)* full decentralization, *(c)* public access, and *(d)* the incorporation of the payment of fees automatically.

Table 1. Comparison of Related Work

Work	Data Storage	Decentralized	Public	Automatic Payments
FENDE [2]	Traditional Database	✗	✗	✗
T-NOVA [16]	Traditional Database	✗	✗	✗
VMOA [3]	Blockchain	✓	✗	✗
SINFONIA [13]	Blockchain	✓	✗	✗
`BUNKER`	Blockchain	✓	✓	✓

3 BC-based Trusted VNF Package Repository

The proposed architecture of BUNKER is depicted in Figure 1. It is composed of two main components: *(i)* the **Graphical User Interface (GUI)**, which is responsible for user interaction, and *(ii)* the **Smart Contract (SC)**, which implements the main systems of BUNKER. It is worth mentioning that the NFV MANO and NFV Infrastructure were not implemented as a third-party solutions are able to provide them. The description of the associated components and their internal systems is presented in details in the following sections. It should be mentioned that BUNKER can be integrated into existing NFV solutions, such as reverse auction mechanisms to find an infrastructure to host VNFs [7] or orchestrators able to manage deployed VNFs [2, 15].

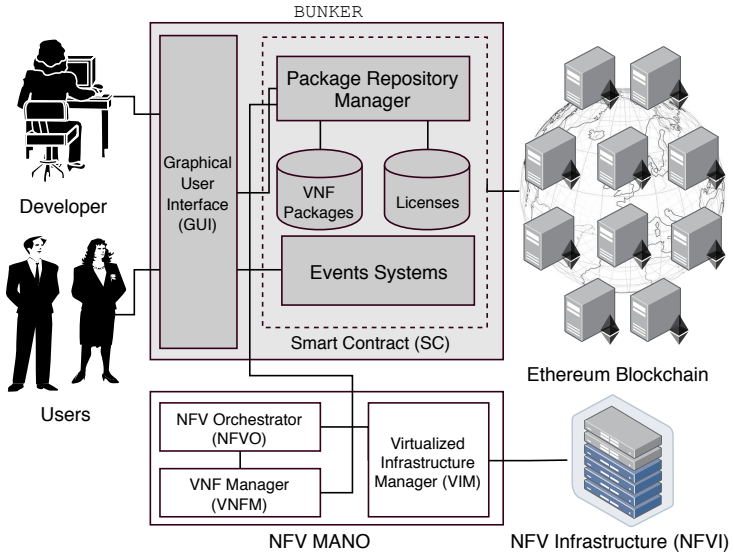

Fig. 1. Proposed BUNKER Architecture

The first component, the **GUI**, is responsible for user interaction and presenting information, such as available and acquired VNF packages, package rating, and prices. Two interaction roles with the **GUI** were identified, *users* and *developers*. Users are able to acquire VNF packages to execute in their NFV environment and submit rates for these packages. Developers are able to offer their VNF packages by registering them in the repository. Moreover, they can delete a VNF from the repository or update its information. To provide these functionalities, the **GUI** implements four systems: *(i) Registration and Upgrade System*, *(ii) Licensing System*, *(iii) Verification System*, and *(iv) Rating System*.

The *Registration and Update System* is used by developers to submit new VNF packages to the repository. Further, developers can maintain their registered VNF packages, *e.g.*, update the package to a new version or to update information that is stored in the repository. Table 2 presents relevant metadata and attributes of VNF packages that can be stored in the repository. In addition, the table contains examples of such attributes. This system allows vendors (*i.e.*, developers) to change attributes and to remove a VNF offering from the repository, and users to retrieve attributes from available VNF packages.

Table 2. VNF Package Information

Metadata	Attributes	Example
Catalog	Package Name	NexGenFirewall
	Description	IPTables-based Firewall with high performance
	Price	1.5 Ether
	Package Link	https://github.com/murielfranco/firewall_repository
	Vendor/Developer	University of Zurich (UZH)
	Category	Protection
	Type	Firewall
	Licensing Type	Monthly
	Version	1.0
VNF	VNF Descriptor	TOSCA standard
	VNF Image	Ubuntu-based
	Requirements	1 vCPU, 4 GB RAM, 6 GB Disk
	Suggested Platform	CloudStack
Other	Vendor Ethereum Address	0x756F45E3FA69347A9A973A725E3C98bC4db0b6a0
	Repository Hash	da6e681320812a87fa7da1416119992da0a1e48e485d2f095ad19872fd6d8e1b
	Business Model	Fixed Price

The *Licensing System* is responsible for handling customers requests to acquire VNF packages. Figure 2 depicts the process of acquiring a package (Function `buy_VNF()` [9]). First, the user requests, by creating a BC transaction, a license of a VNF package through the front end. In the transaction, the customer includes the licensing fee (*i.e.*, package price) and transaction fees. The SC checks whether sufficient funds were included in the request and transfers the licensing fee to the developer. Then, it reads the package's data from the repository, emitting a licensing event containing the necessary information (*e.g.*, package link) to retrieve the VNF and to execute it in the NFV environment. Finally, the front end captures this event and retrieves the package data from the external data storage to be deployed and used in the NFV environment.

Verifying the integrity of the VNF package before deployment and execution is crucial to ensure that its code was not tampered with or its files were not corrupted. Thus, `BUNKER` implements a *Verification System*. This system allows verifying the VNF image's integrity by comparing the hash of the downloaded package with the hash previously generated when the package was added to the repository. Such verification can occur when a new VNF package is acquired, where the system retrieves the package and verifies it against the information stored in the trusted repository, or during *runtime*, because `BUNKER` offers capabilities to re-retrieve the hash and to re-verify the package integrity. This is

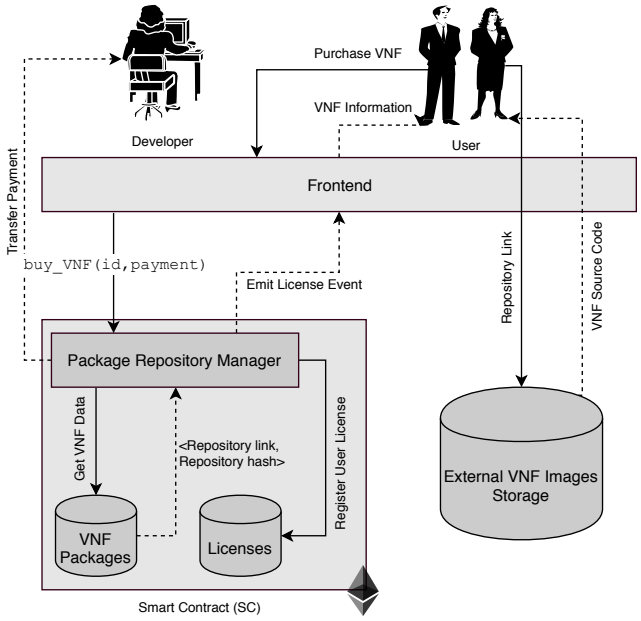

Fig. 2. Data Flow of Acquiring a VNF Package

useful before performing life-cycle operations, such as *upscaling* or *downscaling*, to more instances. Also, it allows retrieving the VNF Descriptor (VNFD) to verify the correctness of configuration and life-cycle operations.

`BUNKER` allows any interested party to register new VNF packages to foster competition. However, there is no curation of the repository's offerings. This leads to a trust issue, as malicious parties may register packages that do not adhere to their specifications. Thus, customers need another way to assess the quality of an offering. For this reason, a *Rating System* was included. It allows licensees to rate a VNF package, providing feedback to future customers. The rating attributes include rating score (*e.g.*, 8.5 out of 10), summary (*e.g.*, VNF executed the promised function), advantages (*e.g.*, quick deployment), and disadvantages (*e.g.*, costly). Unfortunately, language limitations of BC-based SCs pose a challenge to verify the quality of offerings inside SCs. Nevertheless, security verification mechanisms [5] and reputation schemes [8] are planned to be studied to address such a limitation.

The second component, the **SC**, is deployed in the *Ethereum BC*, and implements the functions and data structures necessary to provide a decentralized trusted VNF package repository. This component is composed of the *(i) Package Repository Manager*, *(ii) Events System*, *(iii) VNF Packages* database, and *(iv) Licenses* database [9].

The *Package Repository Manager* is responsible for creating, managing, and maintaining VNF packages entries in the repository. It acts as an intermediate party between the user and repository, receiving all requests (*i.e.*, transactions) to the BC-based repository back end and accessing the repository as necessary. Therefore, this component offers an Application Binary Interface (ABI) for all functions needed in the front end. When a function is called, the repository manager authenticates (relying on the sender's Ethereum address) the user and if authorized, executes the function call and returns the result. This component is implemented as a Soldity SC [9]. Thus, if the front end calls one of the functions, it is executed on nodes in the Ethereum VM (EVM). The output of the functions performed by the nodes is the same across nodes in the BC. This means that the SC code is running in a trusted environment, and it enforces the correct execution of the implemented code before appending the result in the BC.

The *VNF Packages Repository* stores VNF package details (Table 2), acquired licenses, ratings, and verification information (*e.g.*, package hashes). This repository is only accessible through the *repository manager*, and as it is implemented on a BC-based SC, the information included in the repository is stored in the underlying BC network, incurring costs, which increase with the amount of data stored. In practice, this means that the repository data size should be limited to essential information. Thus, only a link to the VNF package location is stored in the BC, and not the VNF package itself. The package code or application must be hosted on an external data storage. Even though storing data externally of the BC introduces trust issues, the verification system included in `BUNKER` allows verifying the integrity of the packages, tackling this issue by storing an immutable hash of the VNF package.

Ethereum-based SCs allow the developers to emit events inside implemented functions. In `BUNKER`, the *Events System* is responsible to manage and emit events. These events are stored in the transaction's log, which is a special data structure in the Ethereum BC [6]. External applications can listen to specific SC events and perform actions upon receiving such events. `BUNKER` takes advantage of events by implementing an event named `License`. This event is emitted once a VNF is acquired and contains information, such as buyer address, VNF image link, and VNF image hash. Thus, the **GUI** constantly listens for this event to present the user with the information about the VNF that he/she acquired. Moreover, other components of the NFV MANO, such as the VNF Manager (VNFM), are able to listen to this `License` event [9] and automatically clone the VNF image to the user's VNF infrastructure and deploy it. In [14], it is presented the interaction of an NFV infrastructure with an SC.

4 Discussion

As described in [12], determining the VNF package integrity is a critical challenge in the setup of a trusted NFV environment. `BUNKER` addresses this challenge successfully without having to rely on an external Trusted Security Orchestrator (TSecO). This mitigates the single point of failure. `BUNKER` is based on an SC

without any access control and management. As such, any interested party can use all the functions provided, given that they pay the fee needed to update the SC's state. This means that the SC is fully distributed and without the need for any dedicated management. On one hand, there is no maintenance cost because no fees have to be collected to keep the SC running. On the other hand, there is a potential for spam and fake entries that do not deliver functions promised or infringe on trademarks and intellectual property.

To provide access control and verification of vendors, `BUNKER` would need to be managed either by a central authority or a consortium. This increases trust in the repository's content, since the manager can verify the authenticity of vendors before any package is registered. Further, such an approach curates the repository's offerings by checking VNF packages for malicious code and verifying that the functionality complies to the specifications. However, the centralized management of the SC and the authorization of participants may be biased and against the intent of `BUNKER`'s distributed nature. The alternative, offloading the management to a consortium, might not mitigate the problem of malicious participants and could still create conflicts of interest. Thus, the current design without an access control and an uncurated repository may face these challenges, but it reflects in full `BUNKER`'s primary goal of removing the need for a central TTP to ensure VNF package's integrity.

5 Summary and Future Work

This paper presented the design of a novel approach for a trusted BC-based VNF package repository, called `BUNKER`. This repository is designed on top of the public Ethereum BC and is implemented as an SC that stores the VNFs information (*e.g.*, VNF package hash) in the BC and allows developers to register, update, or delete VNFs, and to receive the payment of acquired packages automatically. Moreover, users are able to retrieve the content of the repository, acquire, and rate VNF packages. `BUNKER` provides a tamper-proof storage and since it is distributed and executed across many BC nodes, there exists no single point of failure. All these aspects contribute toward `BUNKER`'s primary goal of providing a trusted and available VNF package repository to users and developers without the need for a centralized TTP.

Based on the details presented herein, it can be concluded that the integration of employing BCs and SCs provides for a feasible and trusted VNF package repository. However, there are challenges remaining as discussed in Section 4 to be addressed. Thus, future work includes *(i)* cost evaluations of `BUNKER` interactions, *(ii)* a security analysis (*e.g.*, VNF verification methods and cryptography to secure the repository data), *(iii)* extending the data storage to support a distributed file system, and *(iv)* an integration with an NFV solution. Overall, `BUNKER`as it stands today in its prototype contributes to a better understanding of a BC employment within NFV to secure NFV MANO operations.

References

1. T. Bocek, B. Stiller: Smart Contracts Blockchains in the Wings. In: Digital Marketplaces Unleashed. Springer, 2018, pp. 169–184.
2. L. Bondan, M. F. Franco, L. Marcuzzo, G. Venancio, R. L. Santos, R. J. Pfitscher, E. J. Scheid, B. Stiller, F. De Turck, E. P. Duarte, A. E. Schaeffer-Filho, C. R. P. d. Santos, L. Z. Granville: FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs. In: IEEE Communications Magazine. Vol. 57. IEEE, January 2019, pp. 13–19.
3. N. Bozic, G. Pujolle, S. Secci: Securing Virtual Mchine Orchestration with Blockchains. In: 1st Cyber Security in Networking Conference (CSNet 2017). Rio de Janeiro, Brazil, October 2017, pp. 1–8.
4. V. Buterin: Ethereum White Paper, `https://github.com/ethereum/wiki/wiki/White-Paper` Last access April 23, 2019
5. O. Demir, W. Xiong, F. Zaghloul, J. Szefer: Survey of Approaches for Security Verification of Hardware/Software Systems. Cryptology ePrint Archive, Report 2016/846, 2016, `https://eprint.iacr.org/2016/846` Last access July 4, 2019
6. Ethereum Foundation: Solidity - Solidity 0.58.0 Documentation, `https://solidity.readthedocs.io/` Last access April 28, 2019
7. M. F. Franco, E. J. Scheid, L. Z. Granville, B. Stiller: BRAIN: Blockchain-based Reverse Auction for Infrastructure Supply in Virtual Network Functions-as-a-Service. In: IFIP Networking 2019 (Networking 2019). Warsaw, Poland, May 2019, pp. 1–9.
8. A. Gruhler, B. Rodrigues, B. Stiller: A Reputation Scheme for a Blockchain-based Network Cooperative Defense. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2019). Washington, USA, April 2019, pp. 71–79.
9. M. Keller: Blockchain-based Trusted VNF Package Repository, 2019, `https://github.com/mkllr888/trusted-VNF-repository` Last access July 4, 2019
10. S. Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System, 2009, `https://bitcoin.org/bitcoin.pdf` Last access March 22, 2019
11. Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG): ETSI GS NFV-MAN 001 - V1.1.1 - Network Functions Virtualisation (NFV); Management and Orchestration, 2014, `http://tiny.cc/NFVMANO` Last access April 1, 2019
12. S. Ravidas, S. Lal, I. Oliver, L. Hippelainen: Incorporating Trust in NFV: Addressing the Challenges. In: 20th Conference on Innovations in Clouds, Internet and Networks (ICIN 2017), March 2017, pp. 87–91.
13. G. A. F. Rebello, I. D. Alvarenga, G. de Teleinformatica e Automacão: SINFONIA: Gerenciamento Seguro de Funcoes Virtualizadas de Rede atraves de Corrente de Blocos. In: Anais do I Workshop em Blockchain: Teoria, Tecnologias e Aplicacoes (WBlockchain - SBRC 2018). Vol. 1. SBC, Brasil, May 2018, pp. 0–14.
14. E. J. Scheid, B. Stiller: Leveraging Smart Contracts for Automatic SLA Compensation - The Case of NFV Environment. In: IFIP 12th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2018). IEEE, Munich, Germany, June 2018, pp. 70–74.
15. The Linux Foundation: OPNFV: An Open Platform to Accelerate NFV, `http://tiny.cc/OPNFV` Last access May 17, 2019
16. G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, Y. Rebahi, A. Kourtis: T-NOVA: A Marketplace for Virtualized Network Functions. In: 2014 European Conference on Networks and Communications (EuCNC 2014). Bologna, Italy, June 2014, pp. 1–5.